Public Keys

Arjen K. Lenstra[1], James P. Hughes[2], Maxime Augier[1], Joppe W. Bos[1], Thorsten Kleinjung[1], and Christophe Wachter[1]

[1] EPFL IC LACAL, Station 14, CH-1015 Lausanne, Switzerland
[2] Self, Palo Alto, CA, USA

Abstract. We performed a sanity check of public keys collected on the web and found that the vast majority works as intended. Our main goal was to test the validity of the assumption that different random choices are made each time keys are generated. We found that this is not always the case, resulting in public keys that offer no security. Our conclusion is that generating secure public keys in the real world is challenging. We did *not* study usage of public keys.

Keywords: Sanity check, public keys, (batch) factoring, discrete logarithm, Euclidean algorithm, seeding random number generators.

1 Introduction

Various studies have been conducted to assess the state of the current public key infrastructure, with a focus on X.509 certificates (cf. [3]). Key generation standards for RSA (cf. [23]) have been analysed and found to be satisfactory in [19]. In [11] and [26] (and the references therein) several problems have been identified that are mostly related to the way certificates are used. In this paper we complement previous studies by concentrating on computational and randomness properties of actual public keys, issues that are usually taken for granted.

Compared to the collection of certificates considered in [11], where shared public keys are "not very frequent", we found a much higher fraction of duplicates. We also found public keys that are not related to the Debian OpenSSL vulnerability but that offer no security at all. The existence of such keys may be crypto-folklore, but it was new to us (but see [12]). This is not a disappearing trend, as may be seen by comparing the results in this paper to those reported in [15]. Vulnerabilities of this sort could affect the expectation of security that the public key infrastructure is intended to achieve. We limited our study to collections of public keys and did not consider issues arising while using them.

We summarize our findings, referring to later sections for details. We collected as many openly accessible public keys as possible from the web, while avoiding activities that our system administrators may have frowned upon. In particular we did not capture or analyse any encrypted traffic of digitally signed documents. The set of 11.7 million public keys that we collected contains 6.4 million distinct RSA moduli. The remainder is almost evenly split between ElGamal keys (cf. [8]) and DSA keys (cf. [25]), plus a single ECDSA key (cf. [25]). The frequency of keys blacklisted due to the Debian OpenSSL vulnerability (cf. [28]) is comparable

to [11]; the findings presented below are not related to this vulnerability. All keys were checked for consistency such as compositeness, primality, and (sub)group membership tests. As the sheer number of keys and their provenance precluded extensive cryptanalysis and the sensibility thereof, per key a modest search for obvious weaknesses was carried out as well. These efforts resulted in a small number of inconsistent or weak keys.

A tacit and crucial assumption underlying the security of the public key infrastructure is that during key setup previous random choices are not repeated. In [11,19] public key properties are considered but this issue is not addressed, with [19] nevertheless concluding that

> The entropy of the output distribution [of standardized RSA key generation] is always almost maximal, ... and the outputs are hard to factor if factoring in general is hard.

We do not question the validity of this conclusion, but found that it can only be valid if each output is considered in isolation. When combining outputs the above assumption sometimes fails. Among all types of public keys collected (except ECDSA), we found duplicates with unrelated owners. This is a concern because, if these owners find out, they may breach each other's security. Duplication of keys is more frequent in our collection than in the one from [11].

We also stumbled upon RSA moduli, not affected by the Debian OpenSSL vulnerability, that offer no security. Their secret keys are accessible to anyone who redoes our work. Assuming access to the public key collection, this is straightforward compared to more traditional ways to retrieve RSA secret keys (cf. [4,16]). Figure 1 depicts a simplified sketch of the situation and how it may evolve. Our findings, of which we do not and will not publicly present any evidence, are confirmed by independent similar work (cf. [10]). As shown in Section 3 we used a computational method different from the one used in [10].

Section 2 presents our data collection efforts. Sections 3 and 4 describe the counts and calculations performed for the RSA-related data and for the ElGamal, DSA, and ECDSA data, respectively. Section 5 summarizes our findings.

2 Data collection

Before the data from [7] was generally available, we started collecting public keys from a wide variety of sources, assisted by colleagues and students. We collected *only* public keys, no encrypted data or digitally signed documents (other than digital certificates). This resulted in almost 5.5 million PGP keys and fewer than 0.1 million X.509 certificates. The latter got a boost with [7] and, to a smaller extent, the data from [11]. We did not engage in web crawling, extensive `ssh`-session monitoring, or other data collection activities that may be perceived as intrusive, aggressive, or unethical. Thus, far more data can be collected than we did (see also [15]).

Skipping a description of our attempts to agree on a sufficiently uniform and accessible representation of the data, by November 2011 the counts had settled

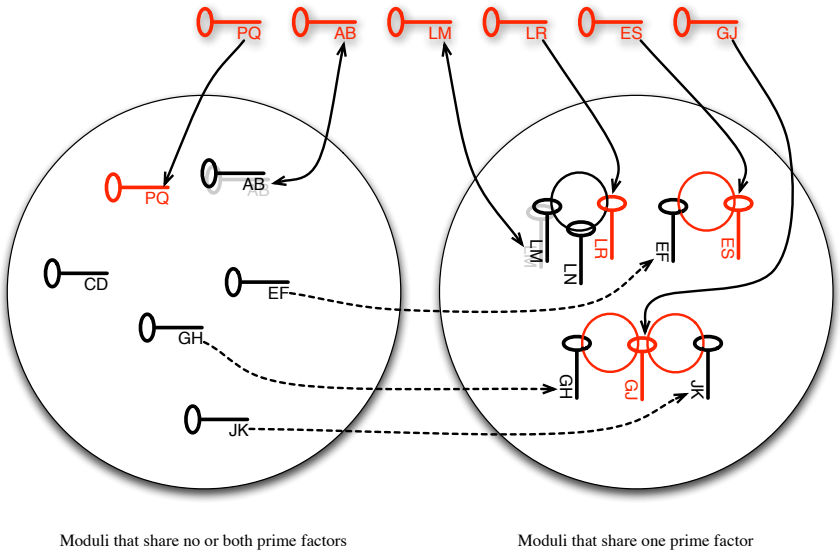

Fig. 1. An existing collection of seven (black) keys is extended with six (red) new keys, where capital letters play the role of (matching) large primes. Initially, keys AB, CD, EF, GH, and JK on the left are secure and keys LM and LN on the right are openly insecure in the same keyring due to the common factor L. New key PQ is secure and appended to the secure list on the left. New key AB duplicates key AB on the left, making both insecure to each other but not to anyone else. New key LM duplicates a key already known to be in the openly insecure group, while key LR results in a new openly insecure modulus on that keyring. Key ES removes known good key EF from the secure keys on the left, resulting in a new openly insecure group on the right consisting of keys EF and ES. Even if the owner of ES now knows that he is insecure and destroys the key, this information can be used by any owners involved to determine the factors of key EF. Key GJ removes two known good keys, GH and JK, from the list of secure keys on the left to form an insecure double keyring on the right (cf. Figure 5 in Section 3). All example keyrings, and many more, occur in the real world. Note that a key that has been dragged from left to right will never be able to return.

as follows: 6 185 372 distinct X.509 certificates (most from the EFF SSL repository, 43 from other sources), and 5 481 332 PGP keys, for a total of at most 11 666 704 public keys. Of the X.509 certificates 6 185 230 are labeled to contain an RSA (modulus, exponent) pair with 141 DSA public keys and a single ECDSA point on the NIST standardized curve `secp384r1` (cf. [2, Section 2.8], [25]). Of the certificates 47.6% have an expiration date later than 2011. About 77.7% of the certifying signatures use SHA1 or better (5287×SHA256, 24×SHA384, 525×SHA512), 22.3% use MD5 (with 122×MD2, 30×GOST, 14×MD4, and 9×RIPEMD160). Both requirements, expiration later than 2011 and usage of SHA1-or-2, are met by 33.4% of the certificates.

Table 1. Most frequently occurring RSA public exponents.

X.509		PGP		Combined	
e	%	e	%	e	%
65537	98.4921	65537	48.8501	65537	95.4933
17	0.7633	17	39.5027	17	3.1035
3	0.3772	41	7.5727	41	0.4574
35	0.1410	19	2.4774	3	0.3578
5	0.1176	257	0.3872	19	0.1506
7	0.0631	23	0.2212	35	0.1339
11	0.0220	11	0.1755	5	0.1111
47	0.0101	3	0.0565	7	0.0596
13	0.0042	21	0.0512	11	0.0313
65535	0.0011	$2^{127}+3$	0.0248	257	0.0241
other	0.0083	other	0.6807	other	0.0774

Of the PGP keys 2 546 752 (46.5%) are labeled as ElGamal public keys, 2 536 959 (46.3%) as DSA public keys, the other 397 621 (7.3%) as RSA public keys. PGP keys have no expiration dates or hashes. All public keys were further analysed as described below.

3 RSA

In this section we present the results of various counts and tests that we conducted on the data labeled as RSA public keys. An RSA public key is a pair (n, e) of a supposedly hard to factor RSA modulus n and a public exponent e. The corresponding secret key is the integer d such that $de \equiv 1 \bmod \varphi(n)$ or, equivalently, the factorization of n.

Public exponents. Table 1 lists the ten most frequent public exponents along with their percentage of occurrence for the RSA keys in the X.509 certificates, the PGP keys, and when combined. Except for eight times $e = 1$ and two even e-values among the PGP RSA keys, there is no reason to suspect that the e-values are not functional. Two e-values were found that, due to their size and random appearance, may correspond to a short secret exponent (we have not investigated this). The public exponents are not further regarded below.

Debian moduli. Two of the n-values, a 1024 and a 2048-bit one each occurring once, were discarded because they could be fully factored using the data from [20] (cf. Debian OpenSSL vulnerability in [28]). A further 30097 n-values (0.48%, with 21459 distinct ones) were found to be blacklisted (cf. [24]), but as their factors were not easily available they were kept.

Shared moduli. We partition the set of 6 185 228 X.509 certificates into *clusters*, where certificates in the same cluster contain the same RSA modulus. There is a considerable number of clusters containing two or more certificates, each of which could be a security issue; clusters consisting of one certificate, on the

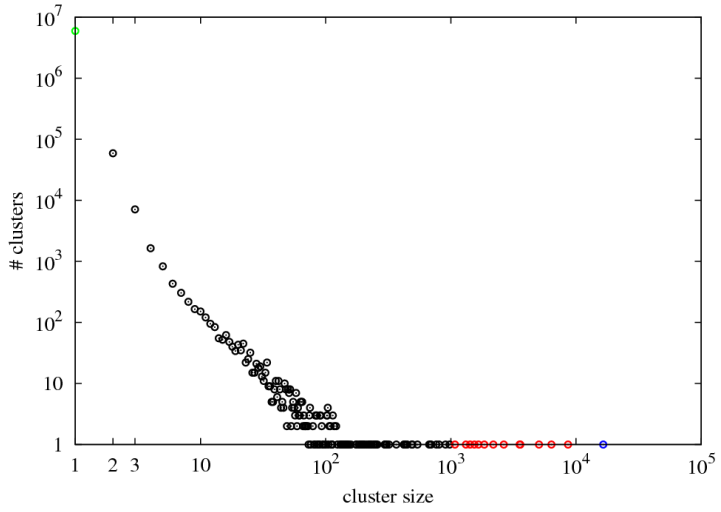

Fig. 2. Number of certificate clusters as a function of the cluster-size.

other hand, are the good cases. As depicted in Figure 2, there is one cluster with 16489 certificates (the blue circle on the x-axis), followed by clusters of sizes 8366, 6351, 5055, 3586, 3538, 2645, for a total of 14 clusters with more than a thousand certificates (the red and blue circles on the x-axis; the 5055 share a blacklisted modulus, with no other blacklisted modulus occurring more than seven times). On the other side of the scale the number of good cases is 5 918 499 (the single green circle on the y-axis), with 58913 and 7108 clusters consisting of two and three certificates, respectively. It follows that $6\,185\,228 - 5\,918\,499 = 266\,729$ X.509 certificates (4.3%) contain an RSA modulus that is shared with another X.509 certificate. With 71024 clusters containing two or more certificates it follows that there are $5\,918\,499 + 71024 = 5\,989\,523$ different n-values.

Looking at the owners with shared n-values among the relevant set of 266 729 X.509 certificates, many of the duplications are re-certifications or other types of unsuspicious recycling of the same key material by its supposedly legal owner. It also becomes clear that any single owner may come in many different guises. On the other hand, there are also many instances where an n-value is shared among seemingly unrelated owners. Distinguishing intentionally shared keys from other duplications (which are prone to fraud) is not straightforward, and is not facilitated by the volume of data we are dealing with (as 266 729 cases have to be considered). We leave it as a subject for further investigation into this "fuzzy" recognition problem to come up with good insights, useful information, and reliable counts.

The 397 621 PGP RSA keys share their moduli to a much smaller extent: one n-value occurs five times and 27 occur twice. Overall, 28 n-values occur more than once, for a total of 59 occurrences. The n-value that occurs in five PGP keys

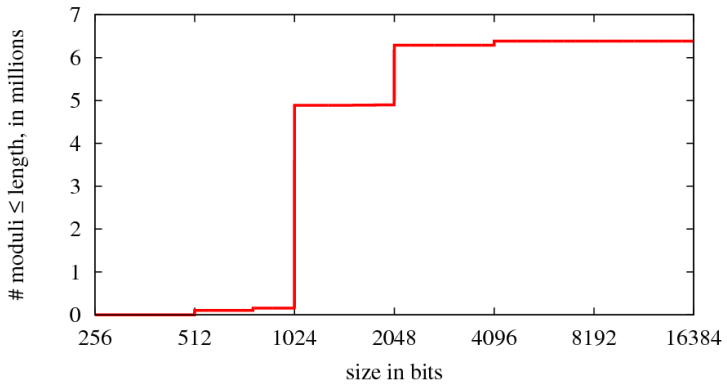

Fig. 3. Cumulative number of modulus sizes for RSA.

also occurs twice among the X.509 certificates, and all seven occurrences refer to the same owner. For some of the other 27 multiple occurrences of n-values unique ownership of the RSA keys was harder to assess.

Distinct moduli. As seen above, we extracted 5 989 523 different n-values from the X.509 certificates. Similarly, $397\,621 - 59 + 28 = 397\,590$ of the PGP n-values are unique. Joining the two sets resulted in 6 386 984 distinct values, with the 129 n-values contained in both sets occurring in 204 X.509 certificates and in 137 PGP keys: as mentioned, some PGP keys are X.509-certified as well (though we have not tried to establish unique or conflicting ownerships, as this already proved to be infeasible for keys shared just among X.509 certificates). In order not to make it easier to re-derive our results, most information below refers to the joined set of unique values, not distinguishing between X.509 and PGP ones.

Modulus sizes. The cumulative sizes of the moduli in the set of 6 386 984 n-values are depicted in Figure 3. Although 512-bit and 768-bit RSA moduli were factored in 1999 (cf. [1]) and 2009 (cf. [13]), respectively, 1.6% of the n-values have 512 bits (with 0.01% of size 384 and smallest size 374 occurring once) and 0.8% of size 768. Those moduli are weak, but still offer marginal security. A large number of the 512-bit ones were certified after the year 2000 and even until a few years ago. With 73.9% the most common size is 1024 bits, followed by 2048 bits with 21.7%. Sizes 3072, 4096, and 8192 contribute 0.04%, 1.5%, and 0.01%, respectively. The largest size is 16384 bits, of which there are 181 (0.003%).

Primality, small factors, and other tests. Two of the unique n-values are prime, 171 have a factor $< 2^{24}$ (with 68 even n-values) after removal of which six cofactors are prime. About 25% of the remaining 165 composites were fully factored after a modest search for small factors using the implementation from [29] of the elliptic curve method (ECM, cf. [17]), some of the others may indeed be hard to factor and could, in principle, serve as RSA modulus. Nevertheless, these 173 n-values do not comply with the standards for the generation of RSA moduli

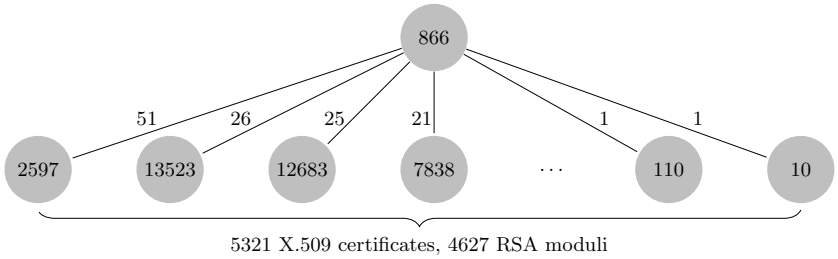

Fig. 4. The largest depth one tree found, on 4628 vertices, with the 512-bit prime p_{866} as root and leaves p_{2597}, p_{13523}, …, p_{10}. The edges correspond to 4627 distinct 1024-bit RSA moduli, with labels indicating the number of distinct X.509 certificates and PGP keys containing the RSA modulus corresponding to the edge, for a total of 5321 certificates. All certificates have expired and use SHA1 as hash function.

(cf. [19]) and they were discarded. Nine cases are probably due to copy-and-paste errors, as eight proper moduli were found that differed from the wrong ones in a few hexadecimal positions (two distinct wrong moduli match up with the same correct one).

Fermat's factorization method, which works well if two factors are close together, did not produce any factors. In particular we found no moduli as reported by Mike Wiener [27] ($n = pq$ with p prime and q the least prime greater than p, and thus with p the largest prime $\leq [\sqrt{n}]$). Neither were there any non-trivial powers.

Moduli with shared factors. Moduli that share one prime factor result in complete loss of security for all moduli involved. We discuss the results based on the graphs spawned by the moduli and shared factors. If different users make different choices during key setup, the graph associated to c distinct n-values (cf. Introduction) would consist of c connected components[3], each consisting of a single edge connecting two unidentified – and supposedly unidentifiable – primes. This turned out not to be the case: it took a matter of hours on a single core to find 1995 connected components that each consist of at least two edges. Much larger datasets can be handled without trouble. Our calculation uses a simple-minded binary tree, forming a parent node $\mathrm{lcm}(a, b)$ for leaves a, b while taking appropriate action if $\gcd(a, b) > 1$ and using the subquadratic multiplication and greatest common divisor implementations from [9]. It scales well and is marginally slower than the more contrived gcd-calculation described in [10] but uses less memory. On a 1.8GHz i7 processor the straightforward approach would require about ten core-years and would not scale well. Inclusion of the p and q-values from Section 4 and the primes from [20] related to the Debian OpenSSL vulnerability [28] did not produce additional results.

[3] Two distinct vertices are in the same connected component if and only if they are connected by a path consisting of edges in the graph.

Table 2. The s-column indicates the number of depth one trees with ℓ leaves for which all edge multiplicities are equal to one, the m-column the number of trees for which at least one edge occurs at least twice, and $T = s + m$ the total. The bold entry corresponds to the depth one tree depicted in Figure 4.

ℓ	s	m	T	ℓ	s	m	T	ℓ	s	m	T	ℓ	s	m	T
2	1009	191	1200	13	3	3	6	24	1	1	2	59	0	1	1
3	259	86	345	14	2	3	5	26	0	1	1	61	0	1	1
4	95	44	139	15	1	4	5	27	0	1	1	63	1	2	3
5	43	32	75	16	2	1	3	32	0	1	1	65	0	1	1
6	23	29	52	17	1	2	3	33	0	1	1	92	0	1	1
7	20	19	39	18	1	1	2	35	0	2	2	121	0	1	1
8	13	17	30	19	1	2	3	36	1	1	2	151	0	1	1
9	4	11	15	20	1	2	3	37	0	1	1	348	0	1	1
10	3	8	11	21	0	3	3	42	0	1	1	379	0	1	1
11	3	9	12	22	1	2	3	44	0	2	2	**4627**	0	**1**	**1**
12	3	3	6	23	0	1	1	46	0	1	1				

Of the 1995 connected components, 1988 are depth one trees[4]. Of those 1200 have two leaves (i.e., 1200 pairs of n-values, each with a distinct prime factor in common), 345 three leaves, etc., up to a single one with 4627 leaves (i.e., 4627 n-values all with the same prime factor in common). It is not uncommon for an n-value corresponding to an edge of these depth one trees to occur more than once as an RSA modulus: 497 of the 1988 depth one trees have at least one edge that corresponds to an RSA modulus that occurs in at least two X.509 certificates or PGP keys. In the other 1491 depth one trees all edge multiplicities are one. Table 2 lists for each number of leaves ℓ how often each type occurs, with the s-column the number of trees for which all n-values occur once as RSA modulus in an X.509 certificate or PGP key, the m-column the number of trees for which at least one n-value occurs as RSA modulus in at least two X.509 certificates or PGP keys, and the total $T = s + m$. For smaller tree-sizes s is larger, for larger trees multiple occurrence of moduli is more common.

Six of the other seven connected components contain four vertices and three edges, but are not depth one trees. Each of these six components thus consists of a "central" n-value that has a factor in common with each of two other, co-prime n-values. The remaining connected component is the most intriguing – or suspicious – as it is a complete graph on nine vertices (K_9): nine primes, each of whose $\binom{9}{2} = 36$ pairwise products occurs as n-value.

Denoting the primes identified with the vertices of the graph by p_1, p_2, ... (using an ordering naturally implied by our representation), Figures 4, 5, and 6 depict the largest depth one tree, the six four-vertex components, and the K_9, respectively, with the edge labels indicating the number of X.509 certificates and PGP keys containing the corresponding n-value as RSA modulus. Note that all moduli in the K_9 occur quite frequently.

[4] A depth one tree has no cycles and contains one *root* vertex with edges leading to all other vertices, as in Figure 4.

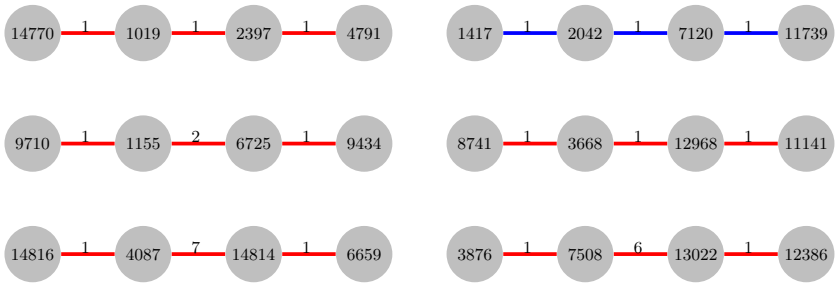

Fig. 5. Six connected components consisting of four vertices, with labels as in Figure 4. The eight primes in the top two components are 512 bits long, the other 16 are 256-bit primes). The red edges correspond to RSA moduli contained in certificates that will not expire anytime soon and that use SHA1 as hash function. The blue ones will expire soon and use MD5.

Any two n-values associated to edges in the same depth one tree can be factored. Two n-values associated to other edges can be factored if the edges are adjacent (i.e., share a vertex), or one finds a path connecting them. For non-adjacent edges in the same connected component from Figure 5 that is the unique central edge, for edges in the K_9 many paths are possible. All required edges are in our set of n-values.

Affected RSA moduli and certificates. The 1995 components contain 14901 vertices and 12934 edges: 14901 distinct primes fully factoring 12934 distinct n-values (0.2% of the total), 11699 of which each occurs as RSA modulus in a single X.509 certificate or PGP key, and 1235 occurring more than once in, in total, 9720 certificates and keys. Thus, $11699 + 9720 = 21419$ X.509 certificates and PGP keys are affected. Note that affected moduli are much more frequently shared than non-affected ones. None of the affected moduli are blacklisted.

Of the primes, 14592 are 512 bits long, 307 are 256-bit, and the remaining two have 257 bits. Of the n-values, 214 are 512 bits long, and there are 12720 of 1024 bits. Of the 512-bit n-values thus factored, 47 occur as RSA moduli in 188 X.509 certificates that have not expired and use SHA1. Of the factored 1024-bit n-values, 3201 occur as RSA moduli in 5250 certificates that have not expired and that use SHA1, of which 617 are regular non-self-signed end-user certificates with "`CA=false`" (with 390 distinct RSA moduli). The majority (4633, with 2811 moduli) has "`CA=true`" and is self-signed, of which 727 (304 moduli) have been used to certify other RSA moduli (none among our collection of affected moduli). These 727 certificates share their "`issuer`"-field and the 304 RSA moduli occur in depth one trees not containing moduli owned by others. So, this security issue is reasonably contained. But 4445 of the 5250 certificates (and 537 of the 617 end-user ones) have no relation to that `issuer` and have a wide variety of "`issuer`" and "`subject`"-fields. This could be a security concern. We do not and will not reveal what types of users or devices are affected. We note, however,

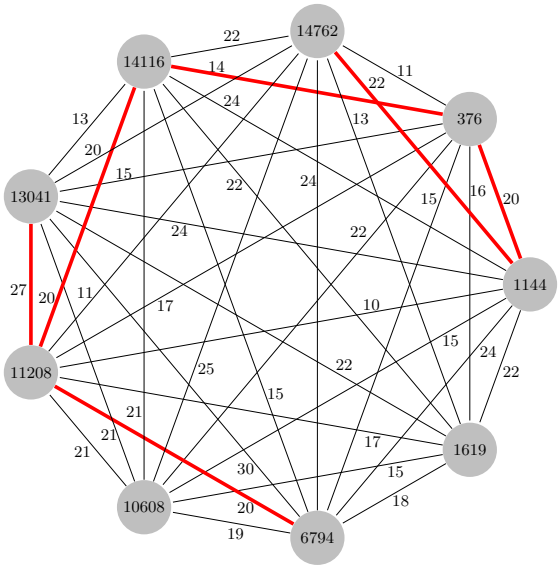

Fig. 6. Connected component consisting of nine vertices, corresponding to primes p_{376}, p_{1144}, ..., p_{14762} (all 512-bit). With labels as in Figure 4, in total 687 X.509 certificates are involved. Six of those certificates have not expired yet, use SHA1 as hash function (as opposed to MD5), and have "`CA=false`"; the red edges correspond to the RSA moduli contained in those six certificates.

that our data give us more reason for concern than reported elsewhere (cf. [10]) and that affected "flesh and blood" users that we talked to were not pleased[5].

Discussion. Generation of a regular RSA modulus consists of finding two random prime numbers. This must be done in such a way that these primes were not selected by anyone else before. The probability not to regenerate a prime is commensurate with the security level if NIST's recommendation [25, page 53] is followed to use a random seed of bit-length twice the intended security level. Clearly, this recommendation is not always followed.

Irrespective of the way primes are selected (additive/sieving methods or methods using fresh random bits for each attempted prime selection), a variety of obvious scenarios is conceivable where poor initial seeding may lead to mishaps, with duplicate keys a consequence if no "fresh" local entropy is used at all. If the latter is used, the outcome may be worse: for instance, a not-properly-random first choice may be prime right away (the probability that this happens is inversely proportional to the length of the prime, and thus non-negligible) and miss its chance to profit from local entropy to become unique. But local entropy may lead to a second prime that is unique, and thus a vulnerable modulus.

The above may, to some extent, explain the occurrence of duplicate RSA moduli and depth one trees. But we cannot explain the relative frequencies

[5] "Donnerwetter!"

and appearance of these mishaps. Neither do we understand how connected components in Figures 5 and 6 may be expected to appear other than by very poor seeding or intentional malfeasance. Also, the great variation of issuers and subjects of the affected X.509 certificates (including the K_9) is disconcerting. No correlation between certification time and vulnerability of keys was detected. Vague, hand-waving arguments suggest that some of the devices involved may have used about 32 bits of entropy.

Avoiding two random choices during RSA modulus generation is straightforward (cf. [14]). But the resulting moduli may have other, as yet unpublished weaknesses (we are not aware of serious ones). It is better to make sure that cryptographic keys are generated only after proper initialization of the source of randomness.

4 ElGamal, DSA, and ECDSA

In this section we present the results of various counts and tests that we conducted on the data labeled as ElGamal, DSA, or ECDSA public keys. In neither collection did we find any of the numbers from [20] (cf. Debian OpenSSL vulnerability [28]).

4.1 ElGamal

An ElGamal public key consists of a triple (p, g, y) where p is prime, g is a generator of the multiplicative group $(\mathbf{Z}/p\mathbf{Z})^*$ or a subgroup thereof of small index, and y is an element of $\langle g \rangle$. The secret key is an integer $x \in \{0, 1, \ldots, p-2\}$ with $g^x = y$.

Correct ElGamal keys. Among the PGP keys, 2 546 752 are labeled as ElGamal public keys. Three are incomplete and were discarded. Of the remaining triples 82 contain a composite p-value, resulting in 2 546 667 triples with correct p-values. Almost half (38) of the wrong p-values share a pattern with 65.6% of the p-values in the correct ElGamal keys, cf. below.

Restricting to the triples (p, g, y) with prime p-values, a triple is a correct ElGamal public key if $y \in \langle g \rangle$. To verify this the order of g, and thus the factorization of $p - 1$, is needed. This is easy for *safe primes* (i.e., primes p for which $(p-1)/2$ is prime), but may be hard otherwise. The order of g could be established for 70.8% of the triples (65.6% with safe primes, 5.2% with primes p for which $(p-1)/(2m)$ is prime and $m > 1$ has only small factors) and could reasonably be guessed for the other 29.2% (almost all with primes p for which $(p-1)/2$ is composite but has no small factors). For at least 16.4% of the ElGamal keys the g-values do not generate $(\mathbf{Z}/p\mathbf{Z})^*$. This led to 33 failed membership tests $y \in \langle g \rangle$, i.e., an insignificant 0.001% of the triples. Note that if $y \in \langle g \rangle$ a secret key exists; it does not follow that the owner knows it. A handful of triples were identified with peculiar y-values for which it is doubtful if a secret key is known to anyone.

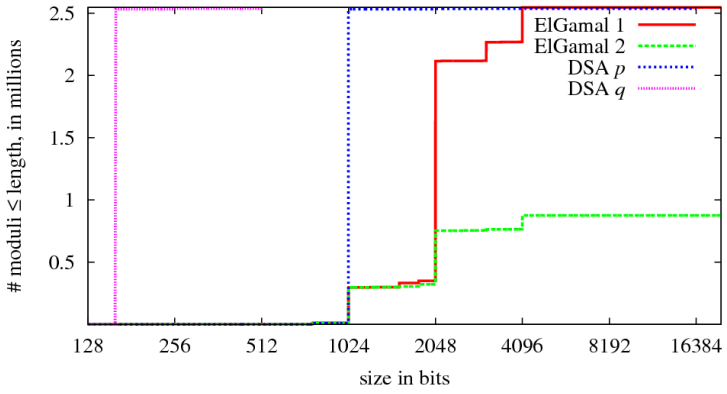

Fig. 7. Cumulative numbers of p and q-sizes in ElGamal and DSA public keys, indicated by "ElGamal 1", "DSA p", and "DSA q"; cumulative sizes of the distinct ElGamal p-values is indicated by "ElGamal 2".

Shared ElGamal keys. Six of the ElGamal keys occur twice: two keys with two unrelated owners each, and four keys occurring twice but with the same owner.

ElGamal key sizes. Figure 7 depicts the cumulative p-sizes in the set of 2 546 628 correct ElGamal keys. There are 1437 different p-sizes, ranging from thrice 256 bits to nine times 16384 and once 20000. Most frequent are 2048 bits (69.3%), 1024 (11.2%), 4096 (10.8%) and 3072 (5.8%) followed by 1536 (1.3%), 1792 (0.7%), 768 (0.4%), and 1025 (0.04%).

Shared primes, generators. Primes and generators may be shared. Among the 2 546 628 distinct ElGamal keys 876 202 distinct p-values (and distinct (p, g)-pairs) occur. Despite this high duplication rate, only 93 distinct p-values occur more than once. The four most frequent p-values are "similar". Let $p(x, L)$ denote the least safe prime $\geq x \bmod 2^L$. There is an integer v such that $p(v, L)$ for L-values 2048, 4096, 3072, 1536 occurs as p-value in 52.4%, 6.5%, 5.6%, and 1% of the ElGamal keys, respectively ($p(v, 1024)$ occurs twice, $p(v + 2^{510}, 512)$ once, and as noted above parts of v also occur in incorrect ElGamal keys). We suspect that these p-values, of different sizes, were generated using similar software and identical random seeding (if any), and from the least significant bit up to the most significant one.

All $p(., L)$-values use $g = 2$, which for $L = 2048$ generates $(\mathbf{Z}/p\mathbf{Z})^*$, but for the others an index two subgroup thereof. Overall, $g = 2$ occurs most frequently (70.8%), followed by $g = 5$ (19.5%), 6 (4.7%), 7 (1.9%), 11 (1.3%), and 13 (0.9%), with a total of 76 distinct g-values. No g-values were found that do not have a large order, but for at least 9.6% of the distinct (p, g)-pairs the g-values do not generate $(\mathbf{Z}/p\mathbf{Z})^*$. We can only give a lower bound because, as pointed out above, we failed to find *any* prime factor of 29.2% of the $(p-1)/2$-values in the ElGamal keys (which turns out to be 87.7% of the distinct $(p-1)/2$-values in the set of

distinct p-values). Thus, we cannot be certain that the corresponding generators were properly chosen; consistent failure of all ECM factoring attempts of these numbers suggests, however, that they were well chosen.

Among the distinct ElGamal keys, all y-values are distinct, which is as expected because distinct (p, g)-pairs have negligible probability to lead to the same y-value (and identical (p, g)-pairs with identical y-values have already been identified and removed). The secret keys, however, may still be the same. But as there is no way to tell if that is the case (for distinct (p, g)-pairs) there is no serious security risk even if they are.

4.2 DSA

A DSA public key is a four-tuple (p, q, g, y) where p and q are primes with q dividing $p-1$, the element g generates an order q subgroup of the multiplicative group $(\mathbf{Z}/p\mathbf{Z})^*$, and y is an element of $\langle g \rangle$. The secret key is the integer $x \in \{0, 1, \ldots, q-1\}$ with $g^x = y$.

Correct DSA keys. Among the PGP keys and X.509 certificates, 2 536 959 and 141 four-tuples, respectively, are labeled as DSA keys. All four-tuples were first checked for correctness, casting them aside at the first test they failed. The tests were conducted in the following order: T1: primality of p; T2: primality of q; T3: divisibility of $p-1$ by q; T4: order of g equals q; and T5: order of y equals q. An insignificant 0.002% (66) of the PGP four-tuples are incorrect with failures 12×T1, 2×T2, 10×T4, and 42×T5 (where T2 failed twice for the same q-value, as it occurred twice). The X.509 DSA four-tuples passed all tests. Some of the failures may be due to transcription errors, as they occur in four-tuples that differ from correct ones in a few hexadecimal positions.

Shared DSA keys. The remaining 2 536 893 PGP DSA keys contain very few duplicates: one key occurs thrice (with possibly double ownership) and two keys occur twice each (each with single ownership), resulting in a total of 2 536 889 distinct PGP DSA keys. Although all 141 X.509 DSA keys are distinct, 95 of them are also among the PGP DSA keys, resulting in a total of $2\,536\,889 + 141 - 95 = 2\,536\,935$ DSA keys. We have not checked ownerships of these 95 duplicate DSA keys.

DSA key sizes. The cumulative p and q-sizes in the set of 2 536 935 DSA keys are depicted in Figure 7. There are nine different q-sizes: all except 0.2% (5012) are 160, with 256, 224, and 232 the most frequent exceptions occurring 4016, 702, and 249 times, respectively. The smallest and largest q-sizes are 160 and 512, the latter with seven occurrences. With 78 different sizes the variation among p-sizes is larger, though nowhere close to the variation among ElGamal p-sizes. All except 0.6% (15457) of the p-sizes are 1024, with 768, 2048, 3072 and 512 the most frequent exceptions with 9733, 3529, 1468 and 519 occurrences, respectively. The smallest and largest p-sizes are 512 and 16384, the latter with a single occurrence.

Shared primes, generators. Distinct DSA keys may contain identical p, q, or g values. In total 2 535 074 distinct p-values occur, with 2 535 037 distinct primes occurring once, 22 occurring twice, five occurring thrice, etc., up to a prime occurring 969 times (note the difference with ElGamal). Not surprisingly (but not necessarily, as the same q-value may give rise to many different p-values), the overall counts and numbers of occurrences are the same for the distinct q-values and the distinct (p, q)-pairs. Although the generator also allows considerable variation, the number of distinct (p, q, g)-triples is the same too. For all except 265 of the unique (p, q, g)-triples, the generator g equals $2^{(p-1)/q}$. We have not been able to determine how the other 265 generators were chosen.

The y-values are all distinct among the distinct DSA keys – given that shared keys were already removed, identical y-values would have been odd indeed. The same remark as above applies concerning identical secret keys.

4.3 ECDSA

The only interesting fact we can report about ECDSA is the surprisingly small number of certificates encountered that contain an ECDSA key (namely, just one), and the small number of certificates signed by ECDSA (one self-signed and a handful of RSA keys). As long as one subscribes to the notion of a standardized curve over a finite field of prime cardinality of a special form, as opposed to a randomly but properly chosen curve over a non-special prime field (cf. [18]), there is nothing wrong with the curve parameters `secp384r1`. It offers (in "theory") about 192 bits of security which makes it, security-wise, comparable to 8000-bit RSA moduli (n) and ElGamal or DSA finite field sizes (p), and 384-bit DSA subgroup sizes (q).

4.4 ElGamal and (EC)DSA.

Not surprisingly, the intersection of the sets of p-values for ElGamal and for DSA is empty. We have not tried hard to retrieve any of the secret exponents, i.e., (for ElGamal and DSA) x-values such that $g^x = y$, but have checked that none is less than 2^{12} in absolute value.

Random nonces in ElGamal and (EC)DSA. Unlike RSA, during signature generation ElGamal and (EC)DSA require a random nonce that should be entirely unpredictable (cf. [8,21,22]). We are not aware of any studies that verify whether or not the nonces are properly chosen (with the notable exception of [5]). Collecting data for such a study requires a much more intrusive type of data collection and may be considered unethical. Note, however, that a mishap in the form of a poorly chosen nonce affects *only* the party that makes the poor choice, but does not affect any other party. In particular the choice of two identical nonces for distinct ElGamal or DSA parameters does not affect anyone but the two users involved.

Discussion. Both for ElGamal and DSA a small number of keys were identified that are shared among unrelated parties. This may be a security concern. Furthermore, there were some ill-formatted keys that cannot be expected to work

and that should be of insignificant security concern. From the point of view of this paper, the main security concern for ElGamal and (EC)DSA is the generation of the random nonce; this is a key usage but not a key generation issue and therefore beyond the scope of this paper.

5 Conclusion

We checked the computational properties of millions of public keys that we collected on the web. The majority does not seem to suffer from obvious weaknesses and can be expected to provide the expected level of security. We found that on the order of 0.003% of public keys is incorrect, which does not seem to be unacceptable. We were surprised, however, by the extent to which public keys are shared among unrelated parties. For ElGamal and DSA sharing is rare, but for RSA the frequency of sharing may be a cause for concern. What surprised us most is that many thousands of 1024-bit RSA moduli, including thousands that are contained in still-valid X.509 certificates, offer no security at all. This may indicate that proper seeding of random number generators is still a problematic issue.

The lack of sophistication of our methods and findings make it hard for us to believe that what we have presented is new, in particular to agencies and parties that are known for their curiosity in such matters. It may shed new light on NIST's 1991 decision to adopt DSA as digital signature standard as opposed to RSA, back then a "public controversy" (cf. [6]); but note the well-known nonce-randomness concerns for ElGamal and (EC)DSA (cf. Section 4.4) and what happens if the nonce is not properly used (cf. [5]).

Factoring one 1024-bit RSA modulus would be historic. Factoring 12720 such moduli is a statistic. The former is still out of reach for the academic community (but anticipated). The latter comes as an unwelcome warning that underscores the difficulty of key generation in the real world.

Acknowledgements

We thank Peter Eckersley from EFF for his invaluable assistance and Don Johnson for pointing out [12] to us. We gratefully acknowledge key collection and other efforts by Benne de Weger, Philippe Joye, Paul Leyland, Yann Schoenberger, Christoph Schuba, Deian Stefan, Fabien Willemin, Maryam Zargari and others that need to remain anonymous. The first author appreciates the legal advice from Mrs. Chardonnens. James P. Hughes participated in this project in his individual capacity without sponsorship of his employer or any other party. This work was supported by the Swiss National Science Foundation under grant number 200020-132160.

References

1. S. Cavallar, B. Dodson, A. K. Lenstra, W. M. Lioen, P. L. Montgomery, B. Murphy, H. te Riele, K. Aardal, J. Gilchrist, G. Guillerm, P. C. Leyland, J. Marchand,

F. Morain, A. Muffett, C. Putnam, C. Putnam, and P. Zimmermann. Factorization of a 512-bit RSA modulus. In B. Preneel, editor, *Eurocrypt 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 1–18. Springer, Heidelberg, 2000.

2. Certicom Research. Standards for efficient cryptography 2: Recommended elliptic curve domain parameters. Standard SEC2, Certicom, 2000.
3. D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, 2008.
4. D. Coppersmith. Modifications to the number field sieve. *Journal of Cryptology*, 6(3):169–180, 1993.
5. Darkmirage. PS3 completely cracked, 2011. `http://www.darkmirage.com/2011/01/06/ps3-completely-cracked/`.
6. Y. Desmedt, P. Landrock, A. K. Lenstra, K. S. McCurley, A. M. Odlyzko, R. A. Rueppel, and M. E. Smid. The Eurocrypt '92 controversial issue: Trapdoor primes and moduli (panel). In R. A. Rueppel, editor, *Eurocrypt '92*, volume 658 of *Lecture Notes in Computer Science*, pages 194–199. Springer, 1993.
7. Electronic Frontier Foundation. EFF SSL Observatory. `https://www.eff.org/observatory`, 2010.
8. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. Blakley and D. Chaum, editors, *Crypto 1984*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, Heidelberg, 1985.
9. Free Software Foundation, Inc. *GMP: The GNU Multiple Precision Arithmetic Library*, 2011. Available at `http://www.gmplib.org/`.
10. N. Heninger. New research: There's no need to panic over factorable keys–just mind your Ps and Qs, 2012. `https://freedom-to-tinker.com/blog/nadiah/new-research-theres-no-need-panic-over-factorable-keys-just-mind-your-ps-and-qs`.
11. R. Holz, L. Braun, N. Kammenhuber, and G. Carle. The SSL landscape: a thorough analysis of the x.509 PKI using active and passive measurements. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, IMC '11, pages 427–444. ACM, 2011.
12. D. B. Johnson. ECC, future resiliency and high security systems. Certicom Whitepaper, 1999. `www.comms.engg.susx.ac.uk/fft/crypto/ECCFut.pdf`.
13. T. Kleinjung, K. Aoki, J. Franke, A. K. Lenstra, E. Thomé, J. W. Bos, P. Gaudry, A. Kruppa, P. L. Montgomery, D. A. Osvik, H. te Riele, A. Timofeev, and P. Zimmermann. Factorization of a 768-bit RSA modulus. In T. Rabin, editor, *Crypto 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 333–350. Springer, Heidelberg, 2010.
14. A. K. Lenstra. Generating RSA moduli with a predetermined portion. In K. Ohta and D. Pei, editors, *Asiacrypt '98*, volume 1514 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 1998.
15. A. K. Lenstra, J. P. Hughes, M. Augier, J. W. Bos, T. Kleinjung, and C. Wachter. Ron was wrong, Whit is right. Cryptology ePrint Archive, Report 2012/064, 2012. `http://eprint.iacr.org/`.
16. A. K. Lenstra and H. W. Lenstra Jr., editors. *The development of the number field sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1993.
17. H. W. Lenstra Jr. Factoring integers with elliptic curves. *Annals of Mathematics*, 126(3):649–673, 1987.
18. M. Lochter and J. Merkle. Elliptic curve cryptography (ECC) brainpool standard curves and curve generation. RFC 5639, 2010.

19. D. Loebenberger and M. Nüsken. Analyzing standards for RSA integers. In A. Nitaj and D. Pointcheval, editors, *Africacrypt '11*, volume 6737 of *Lecture Notes in Computer Science*, pages 260–277. Springer, 2011.
20. H. D. Moore. Debian OpenSSL Predictable PRNG Toys. See `http://digitaloffense.net/tools/debian-openssl/`, 2008.
21. P. Q. Nguyen and I. Shparlinski. The insecurity of the digital signature algorithm with partially known nonces. *Journal of Cryptology*, 15(3):151–176, 2002.
22. P. Q. Nguyen and I. Shparlinski. The insecurity of the elliptic curve digital signature algorithm with partially known nonces. *Design, Codes Cryptography*, 30(2):201–217, 2003.
23. R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.
24. J. Tomášek et al. Blacklisted moduli. See `http://mirror.switch.ch/ftp/mirror/debian/pool/main/o/openssl-blacklist/` and `http://pocitace.tomasek.cz/debian-randomness/index.html`.
25. U.S. Department of Commerce/National Institute of Standards and Technology. Digital Signature Standard (DSS). FIPS-186-3, 2009. `http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf`.
26. N. Vratonjic, J. Freudiger, V. Bindschaedler, and J.-P. Hubaux. The inconvenient truth about web certificates. In *The Workshop on Economics of Information Security (WEIS)*, 2011.
27. M. J. Wiener. Personal communication, 1992.
28. S. Yilek, E. Rescorla, H. Shacham, B. Enright, and S. Savage. When private keys are public: results from the 2008 debian OpenSSL vulnerability. In A. Feldmann and L. Mathy, editors, *Internet Measurement Conference*, pages 15–27. ACM, 2009.
29. P. Zimmermann et al. GMP-ECM (elliptic curve method for integer factorization). `https://gforge.inria.fr/projects/ecm/`, 2012.